UniComal proposals, August 14, 1986:

1. PRINT separators:

When formatting output using normal PRINT statements, it is often desireable to have a separator, which doesn't mean anything. This is not currently available. The default value for ZONE (zero) gives the comma the desired effect. However, when ZONE has been set to a different value, neither comma nor semicolon can be used as a null separator.

This has often lead to poor programming, eg:

```
0010 PROC format( ... ) CLOSED
0020   old_zone:=ZONE
0030   ZONE 0
...
0100   PRINT "HKJHKJH:    ",a," cm."
...
0234   ZONE old_zone
0400 ENDPROC format
```

We also have the impression, that ZONE with a value different from zero is seldom used.

We suggest a change in the meaning of the comma and semicolon separators and a new default value for ZONE:

Old:		New:	
Comma:	Space to ZONE.	Comma:	Do nothing.
Semicolon:	One space.	Semicolon:	Space ZONE.
ZONE:	Default 0	ZONE:	Default 1

If ZONE is not changed in a program, the proposal will have NO influence on the program.

2. IN operator.

As it is now in the standard, A$ IN B$ returns the length of string B$ plus one, if A$ is an empty string. This is due to the fact, that the IN operator has two purposes:

a: To return the position of a string within another string.

b: To find out, if a string is part of another string, true or false.

Due to the lack of a Boolean type in COMAL, 0 is considered as FALSE, and any other value as TRUE. As the empty string is considered part of every string, IN in case b has to return a nonzero value.

In some cases, however, this can be difficult to explain to non-mathematicians - consider the following program:

```
0010 DIM a$ OF 2
0020 PRINT "Do you want to reformat your harddisk? "
0030 REPEAT a$:=KEY$ UNTIL a$ IN "yYnN"
0040 IF a$ IN "yY" THEN PASS "FORMAT C:"
```

When the keyboard buffer is empty, KEY$ returns an empty string. It is difficult to explain, that when you havn't touched the keyboard, you in fact have pressed a key, which belong between the keys: y Y n N.

3. Dyadic multiplication of strings.

Often it is useful to be able to repeat the contents of a string, for instance, if you want a string of 50 A's. We propose the following syntax for that, exemplified as follows:

```
0010 DIM a$ OF 50
0020 a$:=50*"A"
```

The multiplication operator has been expanded to handle an integer and a string operand. The integer operand can be an expression evaluating to a nonnegative value, and the string can be of any length. The order of the integer and the string operand doesn't matter. Zero times any string evaluates to the empty string.

```
0010 FUNC english$(text$,plural)
0020   RETURN plural*("a"+(text$(1) IN "eyuioa")*"n"+" ")+
       (NOT plural)*"the "+text$+plural*"s"
0030 ENDFUNC english$
```